7N-60-CR

79399 Preprint

NAG1 613

P8

CONCURRENT FAULT DETECTION IN THE ATALLAH KOSARAJU DICTIONARY MACHINE[*]

Marc D. Spaulding

Prithviraj Banerjee

Computer Systems Group
Coordinated Science Laboratory
University of Illinois
1101 W. Springfield Av.
Urbana, IL-61801
(217) 333-6564

(NASA-CR-181043) CONCURRENT FAULT DETECTION IN THE ATALLAH KOSARAJU DICTIONARY MACHINE (Illinois Univ.) 8 p Avail: NTIS

N87-70455

Unclas

00/60 0079399

[*] This research was supported in part by the National Aeronautics and Space Administration under Contract NASA NAG 1-613.

1. INTRODUCTION

Recently, a number of tree-structured multiprocessor designs have been proposed [1, 2, 3, 4] for performing operations on a dictionary data structure which consists of a set S of data elements, each of which is composed of a key-record pair $K=(k,r)$. Here k is the key used for searching the data element K and r is the record associated with it. The common operations supported by a dictionary machine are: 'Insert' adds an element to S, 'Delete' removes an existing element from S, 'Member' (or 'Find') determines if a certain element belongs to S, 'Extract-Minimum' and 'Extract-Maximum' determine the elements of S with minimum and maximum key values respectively then remove that element and 'Near' obtains the element with the key value nearest to a given key.

The number of processors needed to support dictionary operations on a large database tends to be quite large if extremely fast processing of instructions is desired. The large number of processors in a system make it more vulnerable to failures, hence it is desirable to have some fault tolerance built into a system. Traditionally, fault tolerance using dynamic redundancy has been viewed as consisting of two parts: 1) fault detection/ location and 2) subsequent reconfiguration of the system around the fault. Most of the work in fault tolerance in tree-structured multiprocessors have addressed the second aspect, namely that of reconfiguring the system after a fault has been detected [5, 6, 7, 8].

The mechanism to detect/locate a faulty processor is assumed to be performed by periodic testing which, unfortunately, is not applicable for detecting transient or intermittent failures. Hence there is a need to build some concurrent fault detection mechanism into complex systems designed for critical applications. In this paper we present a single fault-secure (transient and permanent) version of the dictionary machine proposed by Atallah and Kosaraju [3]. The fault-secure technique assures detection of the failure of any PE before incorrect results are output, given that only one PE fails at once. The addition of fault security to the machine requires that a small amount of hardware be added: one PE, a comparator and the doubling of the number of data registers in the system. The performance of the fault secure machine is not degraded since it maintains the $O(\log n)$ response time (n = number of records stored) and the pipelined nature of the machine.

2. REVIEW OF THE ATALLAH-KOSARAJU DICTIONARY MACHINE

We describe here some of the aspects of the Atallah-Kosaraju dictionary machine in order to make the fault security techniques and the reasons for the required hardware additions clear. For a more extensive description of the machine, the reader is referred to [3].

The machine consists of a binary tree of PE's with communication occurring only between father and son nodes of the tree. Each PE contains 3 registers (left, middle, right) in which key-record (k,r) pairs are stored. The root PE handles all I/O for the machine, since all instruction input and machine responses come through the root node. The PE's are numbered in breadth-first order and each PE is aware of its number. We denote n to be the number of (k,r) pairs currently stored in the machine; this number is known to each PE.

Operations on the tree are handled so as to maintain the (k,r) pairs in a particular configuration. The trees in Fig. 2a show examples of this configuration. A preorder traversal of the tree will always yield a list of the (k,r) pairs sorted in ascending order by key value. Every PE contains either three (k,r) pairs or none, with the exception of the PE with address = $\left\lceil \frac{n}{3} \right\rceil$ which has (n modulo 3) pairs. With this storage system the root PE always contains the minimum and the maximum (k,r) pairs in its left and right data registers respectively. The PE's which are on the same level of the tree as the PE with $p=[n$ modulo 3] key-record pairs constitute the current last level of the tree. PE's below this level do not take part in the machine's operation until more records are added.

Instructions are input at the root in a pipelined fashion and processed by all PE's in a given level at a time. If a response to an instruction is required it is generated by the concerned PE and sent downward in the tree. This response will travel downward until it reaches the last level of the tree, where it will be reflected to travel upward to be output by the root. This procedure preserves the correct order of responses.

Examples of the insertion process are shown in Fig. 2a. In the insertion (deletion) of a (k,r) pair one PE will increase (decrease) its number of (k,r) pairs; this PE will always be on the last level of the tree and is referred to as the INCR (DECR). The rest of the PE's effected by the operation will either move each of their (k,r) pairs into its successor's (next larger key) or predecessor's (next smaller key) position in the tree, depending on the actual situation. The INCR and DECR PE's provide the "slack" in the chain of (k,r) pairs needed to allow insertion or deletion of a (k,r) pair from the tree.

The movement of (k,r) pairs to successor or predecessor positions, in response to a Delete, Insert or Extract instruction, provides the only mechanism through which (k,r) pairs in one subtree of the root may move into the other subtree. It is important to note that such movement requires that the (k,r) pair pass through the middle register of the root as this fact is key in developing a fault secure machine.

3. CONCURRENT FAULT DETECTION SCHEME

The fault secure technique is dependent upon the preorder storage of the (k,r) pairs in the tree and the manner in which that storage scheme is maintained when Inserts, Deletes and Extractions are issued. In order to make the machine fault secure (for 1 PE failure at a time) we add an additional set of three (k,r) pair registers to each PE. Also, we add an additional root PE and a comparator. The extra registers and root PE are used to store a second chain of (k,r) pairs in the tree. The placement of the extra chain is a mirror image of the original chain. Fig. 1 shows the fault secure machine with both chains of (k,r) pairs in evidence. We have essentially two trees of stored (k,r) pairs supported by the same hardware, one the mirror of the other (call them T and TM). For example, PE #2 stores keys 2, 6 and 10 for tree T, and 16,15 and 12 for tree TM, whereas PE #3 stores keys 12, 15, 16 for tree T, and keys 10, 6 and 2 for tree TM. Note that PE #4 and PE #7 have identical middle register contents (=4), PE #4's left register and PE #7's right register are identical as are their right and left registers respectively. The extra root PE serves as the root of the mirror tree TM, and both roots receive identical instruction streams and interact with only one set of the registers contained in PE's 2 and 3 (sons of the roots).

The comparator serves two functions. First, the responses of the two roots are compared continuously. Second, the middle registers of the two roots are compared continuously. Any unequal comparison result indicates that a PE is faulty within the machine; in the first case erroneous outputs are being detected, in the second, erroneous data is being passed from one subtree of the roots to the other.

If a PE in the normal tree T fails, all responses concerning (k,r) pairs in that subtree of the roots are suspect. The mirror tree TM, however, will have all of these same (k,r) pairs stored in the PE's making up the other subtree of the roots. Thus any erroneous response generated because of an erroneous (k,r) pair or PE in T will be compared to a correct response generated by TM and thus detected.

This detection method only breaks down when erroneous (k,r) pairs migrate from one subtree of the roots to the other. In this way one faulty PE may corrupt both of the versions of a (k,r) pair present in the tree. This eventuality is prevented by the comparison of the middle registers of the two roots, through which such migrations must occur. Any such migration in T will be reflected in TM, however, if the (k,r) pair is erroneous in one tree it will not be in the other so the comparator will detect the error. In Figures 2a and 2b we show how this dual corruption by one PE can happen, and is detected.

4. CONCLUSIONS

The hardware we have introduced into the machine has the affect of doubling the time required by each PE to process an instruction. This causes the pipeline interval and response time both to double but does not affect the basic O(log n) nature of the response time, nor the O(1) nature of the pipeline interval found in the original machine. This hardware does not present a large increase in the amount of system hardware because the PE's are quite complex and a large number are likely to be included in any dictionary machine. Thus three extra registers in each PE will not increase system size prohibitively. Lastly, it is necessary to note that no new techniques are required to produce a fault secure version of either of the two redundant instruction handling machines described by Atallah and Kosaraju.

REFERENCES

[1] A.K.Somani and V.K.Agarwal, "An Efficient Unsorted VLSI Dictionary Machine," *IEEE Trans. on Computers*, vol. C-34, no.9, pp. 841-852, September 1985.

[2] T. A. Ottmann, A. L. Rosenberg, and L. J. Stockmeyer, "A Dictionary Machine (for VLSI)," *IEEE Trans. on Computers*, vol. C-31, no.9, pp. 892-897, September 1982.

[3] M. J. Atallah and S. R. Kosaraju, "A Generalized Dictionary Machine for VLSI," *IEEE Trans. on Computers*, vol. C-34, no.2, pp. 151-155, Feb 1985.

[4] A.L.Fisher, "Dictionary Machines with a Small Number of Processors," *11th annual conference on Computer Architecture*, pp. 151-156, 1984.

[5] C. S. Raghavendra, A. Avizienis, and M. Ercegovac, "Fault-Tolerance in Binary Tree Architectures," in *Proc. 13th Int. Symp. on Fault-Tolerant Computing*. pp. 360-364, Jun. 1983.

[6] F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg, "DIOGENES: A Methodology for Designing Fault-Tolerant VLSI Processor Arrays," in *Proc. 13th Int. Symp. on Fault-Tolerant Computing*. pp. 26-32, Jun. 1983.

[7] A. S. M. Hassan and V. K. Agarwal, "A Modular Approach to Fault-Tolerant Binary Tree Architectures," *Proc. 15th Int. Symp. on Fault-Tolerant Computing*, pp. 344-349, June 1985.

[8] B. O. A. Grey, A. Avizienis, and D. A. Rennels, "A Fault-Tolerant Architecture for Network Storage Systems," in *Proc. 14th Int. Symp. on Fault-Tolerant Computing*. pp. 232-239, Jun. 1984.

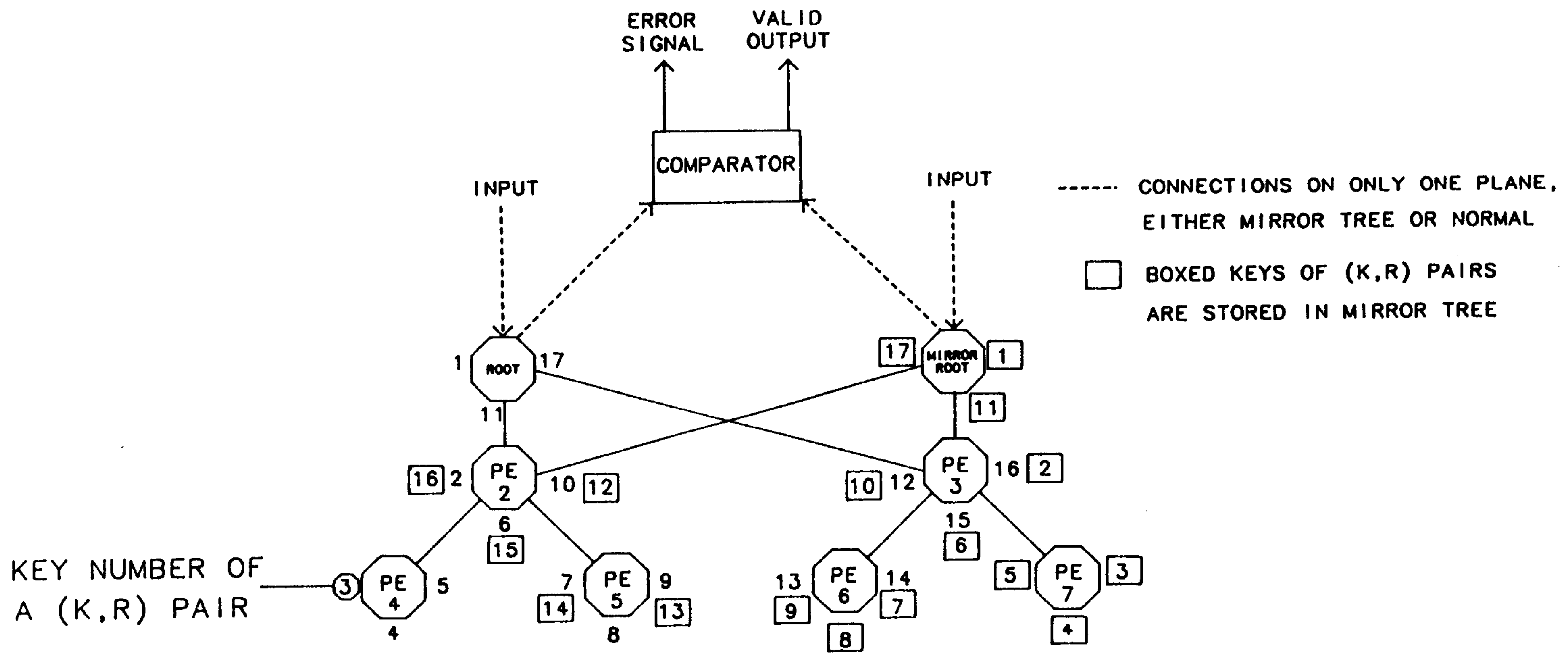
ERROR SIGNAL
VALID OUTPUT
COMPARATOR
INPUT
INPUT
------ CONNECTIONS ON ONLY ONE PLANE, EITHER MIRROR TREE OR NORMAL
BOXED KEYS OF (K,R) PAIRS ARE STORED IN MIRROR TREE
ROOT
MIRROR ROOT
PE 2
PE 3
PE 4
PE 5
PE 6
PE 7
KEY NUMBER OF A (K,R) PAIR
1
17
11
16
2
10
12
6
15
3
5
4
7
14
9
13
8

FIGURE 1. FAULT SECURE MACHINE

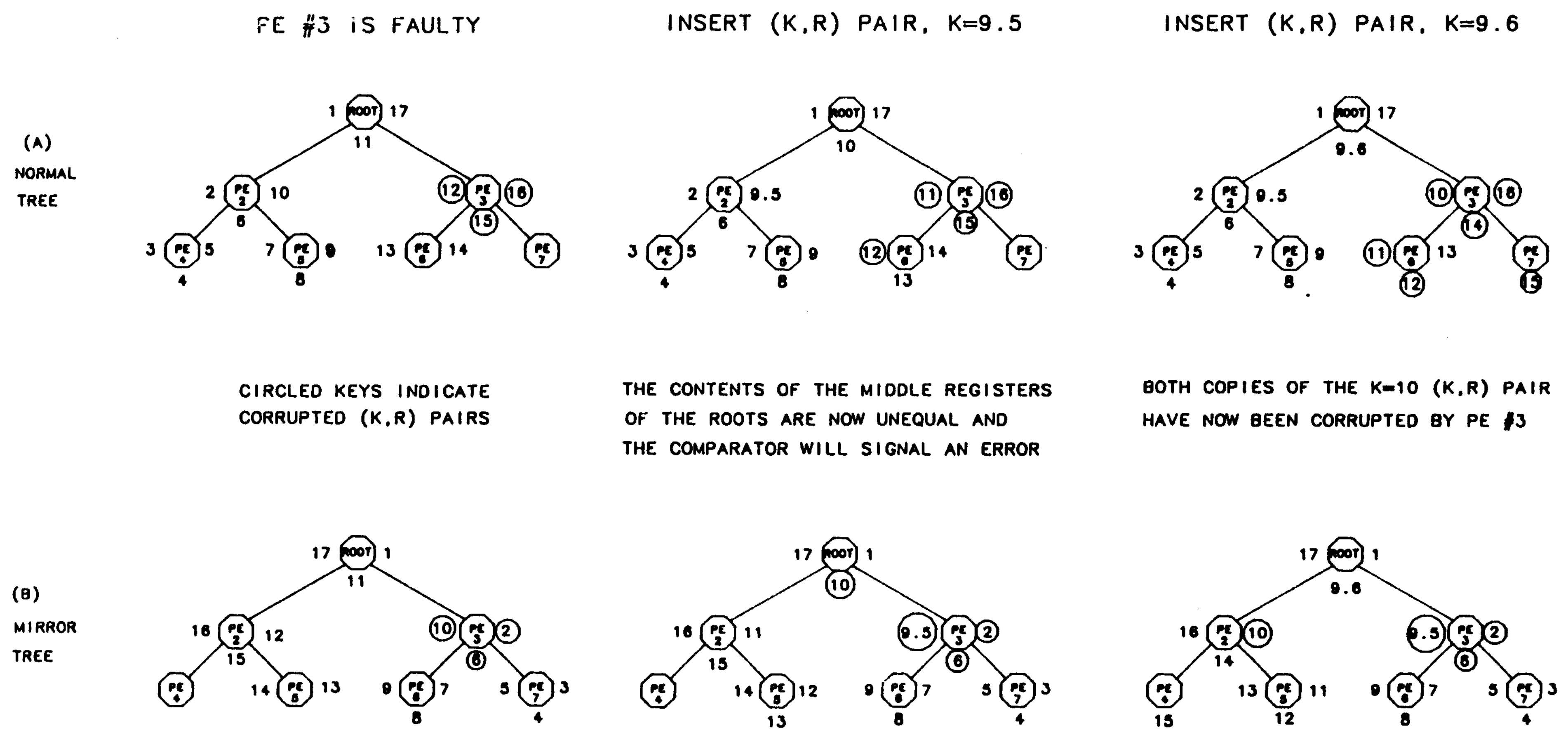
PE #3 IS FAULTY
INSERT (K,R) PAIR, K=9.5
INSERT (K,R) PAIR, K=9.6
(A) NORMAL TREE
(B) MIRROR TREE
ROOT
PE
CIRCLED KEYS INDICATE CORRUPTED (K,R) PAIRS
THE CONTENTS OF THE MIDDLE REGISTERS OF THE ROOTS ARE NOW UNEQUAL AND THE COMPARATOR WILL SIGNAL AN ERROR
BOTH COPIES OF THE K=10 (K,R) PAIR HAVE NOW BEEN CORRUPTED BY PE #3

FIGURE 2. INSERTION OF 2 (K,R) PAIRS INTO THE MACHINE DEMONSTRATES THE CORRUPTION OF DATA IN BOTH SUBTREES BY ONE FAULTY PE, AND ITS DETECTION